По мотивам лекций Е. А. Соколова (ФКН ВШЭ)

Линейная классификация

1 Линейные модели классификации

Мы начнем с задачи бинарной классификации, а многоклассовый случай обсудим позже. Пусть $\mathbb{X} = \mathbb{R}^d$ — пространство объектов, $Y = \{-1, +1\}$ — множество допустимых ответов, $X = \{(x_i, y_i)\}_{i=1}^{\ell}$ — обучающая выборка. Иногда мы будем класс «+1» называть положительным, а класс «−1» — отрицательным.

Линейная модель классификации определяется следующим образом:

$$a(x) = \text{sign}\left(\langle w, x\rangle + w_0\right) = \text{sign}\left(\sum_{j=1}^{d} w_j x_j + w_0\right),$$

где $w \in \mathbb{R}^d$ — вектор весов, $w_0 \in \mathbb{R}$ — сдвиг (bias). Заметим, что функция $\text{sign}(z)$ может выдавать ноль при $z = 0$, но в множество ответов ноль не входит. Во-первых, можем понадеяться, что нулевое значение линейной модели — настолько редкое событие, что нам не придётся иметь с этим дело. Во-вторых, можем считать, что если модель выдала ноль, то она не может выбрать класс и отказывается от классификации — что-то вроде выдачи исключения.

Если не сказано иначе, мы будем считать, что среди признаков есть константа, $x_0 = 1$. В этом случае нет необходимости вводить сдвиг w_0, и линейный классификатор можно задавать как

$$a(x) = \text{sign}\langle w, x\rangle.$$

Геометрически линейный классификатор соответствует гиперплоскости с вектором нормали w. Величина скалярного произведения $\langle w, x\rangle$ пропорциональна расстоянию от гиперплоскости до точки x, а его знак показывает, с какой стороны от гиперплоскости находится данная точка. Таким образом, линейный классификатор разделяет пространство на две части с помощью гиперплоскости, и при этом одно полупространство относит к положительному классу, а другое — к отрицательному.

Напомним, что

$$w_0 + \langle w, x\rangle = 0$$

задает уравнение гиперплоскости (то есть линейное пространство размерности на 1 меньше, чем исходное аффинное точечное пространство).

Расстояние от этой гиперплоскости до произвольной точки z вычисляется по формуле

$$d = \frac{|w_0 + \langle w, z\rangle|}{\|w\|}.$$

w – вектор перпендикулярный гиперплоскости.

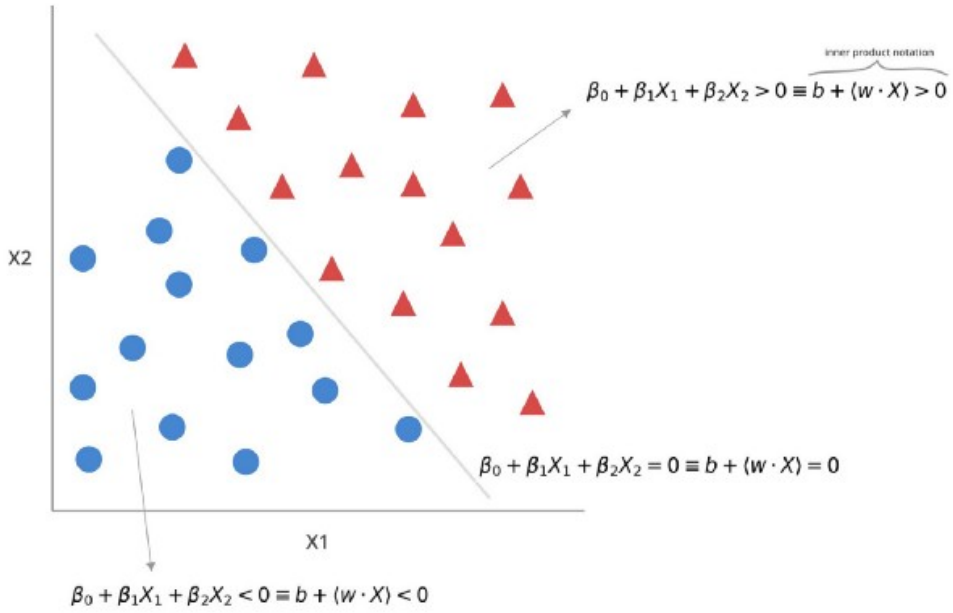


Рис. 1. Разделяющая гиперплоскость.

§1.1 Обучение линейных классификаторов

В задаче регрессии имеется континуум возможных ответов, и при таких условиях достаточно странно требовать полного совпадения ответов модели и истинных ответов — гораздо логичнее говорить об их близости. Более того, как мы выяснили, попытка провести функцию через все обучающие точки легко может привести к переобучению. Способов посчитать близость двух чисел (прогноза и истинного ответа) достаточно много, и поэтому при обсуждении регрессии у нас возникло большое количество функционалов ошибки.

В случае с бинарной классификацией всё гораздо проще: у нас всего два возможных ответа алгоритма и, очевидно, мы хотим видеть как можно больше правильных ответов. Соответствующий функционал называется *долей правильных ответов* (accuracy):

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) = y_i].$$

Здесь используется обозначение

$$[\text{Логическое выражение}] = \begin{cases} 0, & \text{если логическое выражение} = \text{Ложь} \\ 1, & \text{если логическое выражение} = \text{Истина} \end{cases}$$

Нам будет удобнее решать задачу минимизации, поэтому будем вместо этого использовать долю неправильных ответов:

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) \neq y_i] = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}\langle w, x_i \rangle \neq y_i] \to \min_w$$

Этот функционал является дискретным относительно весов, и поэтому искать его минимум с помощью градиентных методов мы не сможем. Более того, у данного

функционала может быть много глобальных минимумов — вполне может оказаться, что существует много способов добиться оптимального количества ошибок. Чтобы не пытаться решать все эти проблемы, попытаемся свести задачу к минимизации гладкого функционала.

Отступы. Заметим, что функционал ошибки можно несколько видоизменить:

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\underbrace{y_i \langle w, x_i \rangle}_{M_i} < 0] \to \min_w$$

Именно для этого в качестве классов выбираются $\{-1, +1\}$, а не $\{0, 1\}$.

Здесь возникла очень важная величина

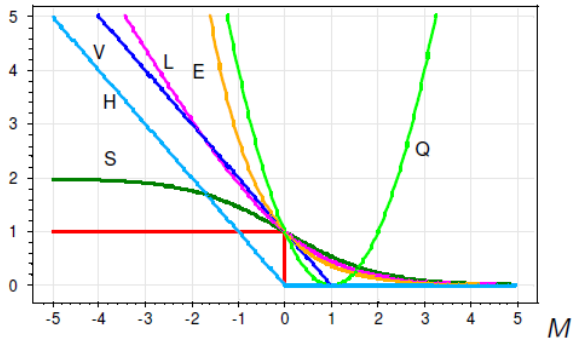
$$M_i = y_i \langle w, x_i \rangle,$$

называемая *отступом* (margin).

Знак отступа говорит о корректности ответа классификатора (положительный отступ соответствует правильному ответу, отрицательный — неправильному), а его абсолютная величина характеризует степень уверенности классификатора в своём ответе.

Напомним, что скалярное произведение $\langle w, x \rangle$ пропорционально расстоянию от разделяющей гиперплоскости до объекта; соответственно, чем ближе отступ к нулю, тем ближе объект к границе классов, тем ниже уверенность в его принадлежности.

Часто используемые непрерывные функции потерь $\mathscr{L}(M)$:



$V(M) = (1-M)_+$ — кусочно-линейная (SVM);
$H(M) = (-M)_+$ — кусочно-линейная (Hebb's rule);
$L(M) = \log_2(1+e^{-M})$ — логарифмическая (LR);
$Q(M) = (1-M)^2$ — квадратичная (FLD);
$S(M) = 2(1+e^{M})^{-1}$ — сигмоидная (ANN);
$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$[M < 0]$ — пороговая функция потерь.

Рис. 2. Верхние оценки на пороговую функцию потерь.

Верхние оценки. Функционал Q оценивает ошибку алгоритма на объекте x с помощью пороговой функции потерь $L(M) = [M < 0]$, где аргументом функции является отступ $M = y\langle w, x\rangle$. Оценим эту функцию сверху во всех точках M кроме, может быть, небольшой полуокрестности левее нуля (см. рис. 2):

$$L(M) \leqslant \tilde{L}(M).$$

После этого можно получить верхнюю оценку на функционал:

$$Q(a, X) \leqslant \frac{1}{\ell}\sum_{i=1}^{\ell} \tilde{L}(y_i\langle w, x_i\rangle) \to \min_w$$

Если верхняя оценка $\tilde{L}(M)$ является гладкой, то и данная верхняя оценка будет гладкой. В этом случае её можно будет минимизировать с помощью, например, градиентного спуска. Если верхнюю оценку удастся приблизить к нулю, то и доля неправильных ответов тоже будет близка к нулю.

Приведём несколько примеров верхних оценок:

1. $\tilde{L}(M) = \log\left(1+e^{-M}\right)$ — логистическая функция потерь
2. $\tilde{L}(M) = (1-M)_+ = \max(0, 1-M)$ — кусочно-линейная функция потерь (используется в методе опорных векторов)
3. $\tilde{L}(M) = (-M)_+ = \max(0, -M)$ — кусочно-линейная функция потерь (соответствует персептрону Розенблатта)
4. $\tilde{L}(M) = e^{-M}$ — экспоненциальная функция потерь

5. $\tilde{L}(M) = 2/(1+e^M)$ — сигмоидная функция потерь

Любая из них подойдёт для обучения линейного классификатора. Позже мы подробно изучим некоторые из них и выясним, какими свойствами они обладают.

2 Логистическая регрессия

§2.1 Оценивание вероятностей

Метод обучения, которые получается при использовании логистической функции потерь

$$\tilde{L}(M) = \log\left(1 + e^{-M}\right),$$

называется логистической регрессией.

Основным его свойством является тот факт, что он корректно оценивает вероятность принадлежности объекта к каждому из классов.

Пусть в каждой точке пространства объектов $x \in \mathbb{X}$ задана вероятность

$$p(y = +1 \,|\, x)$$

того, что объект x будет принадлежать классу $+1$.

Это означает, что мы допускаем наличие в выборке нескольких объектов с одинаковым признаковым описанием, но с разными значениями целевой переменной; причём если устремить количество объекта x в выборке к бесконечности, то доля положительных объектов среди них будет стремиться к $p(y = +1 \,|\, x)$.

Примером может служить задача предсказания кликов по рекламным баннерам. При посещении одного и того же сайта один и тот же пользователь может как кликнуть, так и не кликнуть по одному и тому же баннеру, из-за чего в выборке могут появиться одинаковые объекты с разными ответами. При этом важно, чтобы классификатор предсказывал именно вероятности классов — если домножить вероятность первого класса на сумму, которую заплатит заказчик в случае клика, то мы получим матожидание прибыли при показе этого баннера. На основе таких матожиданий можно построить алгоритм, выбирающий баннеры для показа пользователю.

Итак, рассмотрим точку x пространства объектов. Как мы договорились, в ней имеется распределение на ответах $p(y = +1 \,|\, x)$. Допустим, алгоритм $b(x)$ возвращает числа из отрезка $[0, 1]$. Наша задача — выбрать для него такую процедуру обучения, что в точке x ему будет оптимально выдавать число $p(y = +1 \,|\, x)$. Если в выборке объект x встречается n раз с ответами $\{y_1, \ldots, y_n\}$, то получаем следующее требование:

$$\arg\min_{b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, b) \approx p(y = +1 \,|\, x).$$

При стремлении n к бесконечности получим, что функционал стремится к матожиданию ошибки:

$$\arg\min_{b \in \mathbb{R}} \mathbb{E}\left[L(y, b) \,|\, x\right] = p(y = +1 \,|\, x).$$

Упражнение. Показать, что этим свойством обладает, например, квадратичная функция потерь $L(y, z) = (y - z)^2$, если в ней для положительных объектов использовать истинную метку $y = 1$, а для отрицательных брать $y = 0$.

Упражнение. Примером функции потерь, которая не позволяет оценивать вероятности, является модуль отклонения $L(y,x) = |y - z|$. Можно показать, что с точки зрения данной функции оптимальным ответом всегда будет либо ноль, либо единица.

Это требование можно воспринимать более просто. Пусть один и тот же объект встречается в выборке 1000 раз, из которых 100 раз он относится к классу $+1$, и 900 раз — к классу -1. Поскольку это один и тот же объект, классификатор должен выдавать один ответ для каждого из тысячи случаев. Можно оценить матожидание функции потерь в данной точке по 1000 примеров при прогнозе b:

$$\mathbb{E}\Big[L(y,b)\,|\,x\Big] \approx \frac{100}{1000}L(1,b) + \frac{900}{1000}L(-1,b).$$

Наше требование, по сути, означает, что оптимальный ответ с точки зрения этой оценки должен быть равен $1/10$:

$$\arg\min_{b\in\mathbb{R}}\left(\frac{100}{1000}L(1,b) + \frac{900}{1000}L(-1,b)\right) = \frac{1}{10}.$$

§2.2 Правдоподобие и логистические потери

Хотя квадратичная функция потерь и приводит к корректному оцениванию вероятностей, она не очень хорошо подходит для решения задачи классификации. Причиной этому в том числе являются и слишком низкие штрафы за ошибку — так, если объект положительный, а модель выдает для него вероятность первого класса $b(x) = 0$, то штраф за это равен всего лишь единице: $(1-0)^2 = 1$.

Попробуем сконструировать функцию потерь из других соображений. Если алгоритм $b(x) \in [0,1]$ действительно выдает вероятности, то они должны согласовываться с выборкой.

С точки зрения алгоритма вероятность того, что в выборке встретится объект x_i с классом y_i, равна

$$b(x_i)^{[y_i=+1]}(1 - b(x_i))^{[y_i=-1]}.$$

Исходя из этого, можно записать правдоподобие выборки (т.е. вероятность получить такую выборку с точки зрения алгоритма):

$$Q(a,X) = \prod_{i=1}^{\ell} b(x_i)^{[y_i=+1]}(1 - b(x_i))^{[y_i=-1]}.$$

Данное правдоподобие можно использовать как функционал для обучения алгоритма — с той лишь оговоркой, что удобнее оптимизировать его логарифм:

$$-\sum_{i=1}^{\ell}\big([y_i = +1]\log b(x_i) + [y_i = -1]\log(1 - b(x_i))\big) \to \min$$

Данная функция потерь называется логарифмической (log-loss). Покажем, что она также позволяет корректно предсказывать вероятности. Запишем матожидание

функции потерь в точке x:

$$\mathbb{E}\Big[L(y,b)\,|\,x\Big] = \mathbb{E}\Big[-[y=+1]\log b - [y=-1]\log(1-b)\,|\,x\Big] = \\ = -p(y=+1\,|\,x)\log b - (1-p(y=+1\,|\,x))\log(1-b).$$

Продифференцируем по b:

$$\frac{\partial}{\partial b}\mathbb{E}\Big[L(y,b)\,|\,x\Big] = -\frac{p(y=+1\,|\,x)}{b} + \frac{1-p(y=+1\,|\,x)}{1-b} = 0.$$

Легко видеть, что оптимальный ответ алгоритма равен вероятности положительного класса:

$$b_* = p(y=+1\,|\,x).$$

§2.3 Логистическая регрессия

Везде выше мы требовали, чтобы алгоритм $b(x)$ возвращал числа из отрезка $[0,1]$. Этого легко достичь, если положить $b(x) = \sigma(\langle w,x\rangle)$, где в качестве σ может выступать любая монотонно неубывающая функция с областью значений $[0,1]$. Мы будем использовать сигмоидную функцию: $\sigma(z) = \frac{1}{1+\exp(-z)}$. Таким образом, чем больше скалярное произведение $\langle w,x\rangle$, тем больше будет предсказанная вероятность. Как при этом можно интерпретировать данное скалярное произведение? Чтобы ответить на этот вопрос, преобразуем уравнение

$$p(y=1\,|\,x) = \frac{1}{1+\exp(-\langle w,x\rangle)}.$$

Выражая из него скалярное произведение, получим

$$\langle w,x\rangle = \log\frac{p(y=+1\,|\,x)}{p(y=-1\,|\,x)}.$$

Получим, что скалярное произведение будет равно логарифму отношения вероятностей классов (log-odds).

Как уже упоминалось выше, при использовании квадратичной функции потерь алгоритм будет пытаться предсказывать вероятности, но данная функция потерь является далеко не самой лучшей, поскольку слабо штрафует за грубые ошибки. Логарифмическая функция потерь подходит гораздо лучше, поскольку не позволяет алгоритму сильно ошибаться в вероятностях.

Подставим трансформированный ответ линейной модели в логарифмическую функцию потерь:

$$-\sum_{i=1}^{\ell}\left([y_i=+1]\log\frac{1}{1+\exp(-\langle w,x_i\rangle)} + [y_i=-1]\log\frac{\exp(-\langle w,x_i\rangle)}{1+\exp(-\langle w,x_i\rangle)}\right) = \\ = -\sum_{i=1}^{\ell}\left([y_i=+1]\log\frac{1}{1+\exp(-\langle w,x_i\rangle)} + [y_i=-1]\log\frac{1}{1+\exp(\langle w,x_i\rangle)}\right) = \\ = \sum_{i=1}^{\ell}\log\left(1+\exp(-y_i\langle w,x_i\rangle)\right).$$

Полученная функция в точности представляет собой логистические потери, упомянутые в начале. Линейная модель классификации, настроенная путём минимизации данного функционала, называется логистической регрессией. Как видно из приведенных рассуждений, она оптимизирует правдоподобие выборки и дает корректные оценки вероятности принадлежности к положительному классу.

3 Метрики качества классификации

Выше мы разобрали способ сведения задачи обучения линейного классификатора к минимизации гладкого функционала. При этом часто возникает необходимость в изучении различных аспектов качества уже обученного классификатора. Обсудим подробнее распространенные подходы к измерению качества таких моделей.

Будем считать, что классификатор имеет вид

$$a(x) = \operatorname{sign}(b(x) - t) = 2[b(x) > t] - 1.$$

Линейная модель имеет именно такую форму, если положить $b(x) = \langle w, x\rangle$ и $t = 0$.

§3.1 Доля правильных ответов

Наиболее очевидной мерой качества в задаче классификации является доля правильных ответов (accuracy), которую мы уже упоминали:

$$\operatorname{accuracy}(a, x) = \frac{1}{\ell}\sum_{i=1}^{\ell}[a(x_i) = y_i].$$

Данная метрика, однако, имеет существенный недостаток. Если взять порог t меньше минимального значения прогноза $b(x)$ на выборке или больше максимального значения, то доля правильных ответов будет равна доле положительных и отрицательных ответов соответственно.

Таким образом, если в выборке 950 отрицательных и 50 положительных объектов, то при тривиальном пороге $t = \max_i b(x_i)$ мы получим долю правильных ответов 0.95. Это означает, что доля положительных ответов сама по себе не несет никакой информации о качестве работы алгоритма $a(x)$, и вместе с ней следует анализировать соотношение классов в выборке. Также полезно вместе с долей правильных ответов вычислять *базовую долю* — долю правильных ответов алгоритма, всегда выдающего наиболее мощный класс.

Отметим, что при сравнении различных методов машинного обучения принято сообщать относительное уменьшение ошибки. Рассмотрим два алгоритма a_1 и a_2 с долями правильных ответов r_1 и r_2 соответственно, причем $r_2 > r_1$. Относительным уменьшением ошибки алгоритма a_2 называется величина

$$\frac{(1 - r_1) - (1 - r_2)}{1 - r_1}.$$

Если доля ошибок была улучшена с 20% до 10%, то относительное улучшение составляет 50%. Если доля ошибок была улучшена с 50% до 25%, то относительное

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False negative (FN)	True Negative (TN)

Таблица 1. Матрица ошибок

улучшение также равно 50%, хотя данный прирост кажется более существенным. Если же доля ошибок была улучшена с 0.1% до 0.01%, то относительное улучшение составляет 90%, что совершенно не соответствует здравому смыслу.

§3.2 Матрица ошибок

Выше мы убедились, что в случае с несбалансированными классами одной доли правильных ответов недостаточно — необходима еще одна метрика качества. В данном разделе мы рассмотрим другую, более информативную пару критериев.

Введем сначала понятие матрицы ошибок. Это способ разбить объекты на четыре категории в зависимости от комбинации истинного ответа и ответа алгоритма (см. таблицу 1). Через элементы этой матрицы можно, например, выразить долю правильных ответов:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}.$$

Гораздо более информативными критериями являются *точность* (precision)

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}};$$

и *полнота* (recall):

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Точность показывает, какая доля объектов, выделенных классификатором как положительные, действительно является положительными. Полнота показывает, какая часть положительных объектов была выделена классификатором.

Рассмотрим, например, задачу предсказания реакции клиента банка на звонок с предложением кредита. Ответ $y = 1$ означает, что клиент возьмет кредит после рекламного звонка, ответ $y = -1$ — что не возьмет. Соответственно, планируется обзванивать только тех клиентов, для которых классификатор $a(x)$ вернет ответ 1. Если классификатор имеет высокую точность, то практически все клиенты, которым будет сделано предложение, откликнутся на него. Если классификатор имеет высокую полноту, то предложение будет сделано практически всем клиентам, которые готовы откликнуться на него. Как правило, можно регулировать точность и полноту, изменяя порог t в классификаторе $a(x) = \text{sign}(b(x) - t) = 2[b(x) > t] - 1$. Если выбрать t большим, то классификатор будет относить к положительному классу небольшое число объектов, что приведет к высокой точности и низкой полноте. По мере уменьшения t точность будет падать, а полнота увеличиваться. Конкретное значение порога выбирается согласно пожеланиям заказчика.

Отметим, что точность и полнота не зависят от соотношения размеров классов. Даже если объектов положительного класса на порядки меньше, чем объектов отрицательного класса, данные показатели будут корректно отражать качество работы алгоритма.

Существует несколько способов получить один критерий качества на основе точности и полноты. Один из них — F-мера, гармоническое среднее точности и полноты:

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

Среднее гармоническое обладает важным свойством — оно близко к нулю, если хотя бы один из аргументов близок к нулю. Именно поэтому оно является более предпочтительным, чем среднее арифметическое (если алгоритм будет относить все объекты к положительному классу, то он будет иметь recall $= 1$ и precision $\ll 1$, а их среднее арифметическое будет больше $1/2$, что недопустимо). Можно заметить, что F-мера является сглаженной версией минимума из точности и полноты.